nidmviewer Documentation

Release 1.0

Vanessa Sochat

Jul 05, 2018

Contents

1 Installation **3**

1.1 Running Examples 3

2 nidmviewer **5**

2.1 nidmviewer package 5

3 Indices and tables **13**

Python Module Index **15**

This is a command line (to browser) as well as python module for rendering NIDM Results (turtle files) into an interactive viewer. The viewer currently supports parsing and interactive browsing of peak coordinates, viewing associated brain maps, and saving static images to file.

Contents:

CHAPTER 1

Installation

To install

```
pip install git+git://github.com/vsoch/nidmviewer.git
```

1.1 Running Examples

1.1.1 Python (server)

```
#!/usr/bin/python

from nidmviewer.viewer import generate
from glob import glob
import os

# HTML FOR EMBEDDING #####################################################
ttl_files = glob("fsl/*.ttl")
html_snippet = generate(ttl_files=ttl_files)

# LOCAL BROWSER ##########################################################
httpd = generate(ttl_files=nidm_files,base_image=standard_brain,view_in_browser=True)
```

1.1.2 Command Line

When installing with setup.py, an executable, *nidmviewer* is installed in your bin to view nidm files on the fly:

```
nidmviewer fsl/nidm.ttl
```

You can see the basic usage by typing the command:

```
nidmviewer

usage: nidmviewer [-h] [--columns_to_remove COLUMNS_TO_REMOVE] ttl base
nidmviewer: error: too few arguments
usage: nidmviewer [-h] [--columns_to_remove COLUMNS_TO_REMOVE] ttl base

command line or server tool to view or compare nidm results.

positional arguments:

    ttl                    List of comma separated ttl files to parse.
    base                   base image (standard brain map) to use for the viewer
                           background.
```

CHAPTER 2

nidmviewer

2.1 nidmviewer package

2.1.1 Subpackages

2.1.2 Submodules

2.1.3 nidmviewer.browser module

browser.py: part of nidmviewer package Functions to visualize in browser `nidmviewer.browser.`**`get_svg_html`**(*mpl_figures*)

`nidmviewer.browser.`**`internal_view`**(*html_snippet*, *tmp_file*)

`nidmviewer.browser.`**`run_webserver`**(*port=8088*, *html_page='index.html'*)

`nidmviewer.browser.`**`view`**(*html_snippet*, *copy_list*, *port*)

`nidmviewer.browser.`**`write_file`**(*html_snippet*, *tmp_file*)

2.1.4 nidmviewer.convert module

convert Functions to convert/parse output from nidm sparql queries

`nidmviewer.convert.`**`getjson`**(*nidm_file*, *format='n3'*)

`nidmviewer.convert.`**`parse_coordinates`**(*coordinates*)

> convert a list of xyz strings in format ‘[x,y,z]’ to separate variables This is what we get from the sparql query Parameters ========== coordinates: list
>
> > a list of xyz coordinate strings, each a list in a string ‘[x,y,z]’
>
> coordinate_df: pandas dataframe columns are x,y,z

`nidmviewer.convert.`**`prettyjson`**(*nidm_file*, *format='n3'*)

2.1.5 nidmviewer.scripts module

script.py: part of nidmviewer package Functions to visualize in browser `nidmviewer.scripts.`**`main`**`()`

2.1.6 nidmviewer.sparql module

sparql.py: part of the nidmviewer package Sparql queries

`nidmviewer.sparql.`**`do_query`**`(`*ttl_file*, *query*, *rdf_format='turtle'*, *serialize_format='csv'*, *output_df=True*`)`

`nidmviewer.sparql.`**`get_coordinates`**`(`*ttl_file*`)`

`nidmviewer.sparql.`**`get_coordinates_and_maps`**`(`*ttl_file*`)`

2.1.7 nidmviewer.templates module

templates.py: part of the nidmviewer package Functions to work with html templates

`nidmviewer.templates.`**`add_javascript_function`**(*function_code*, *template*)

`nidmviewer.templates.`**`add_string`**(*tag*, *substitution*, *template*)

`nidmviewer.templates.`**`get_image`**(*image_name*)

`nidmviewer.templates.`**`get_template`**(*html_name*)

`nidmviewer.templates.`**`read_template`**(*html_name*)

`nidmviewer.templates.`**`remove_resources`**(*html_snippet*, *script_names*)

`nidmviewer.templates.`**`save_template`**(*html_snippet*, *output_file*)

2.1.8 nidmviewer.utils module

utils.py: part of the nidmviewer package Functions to work with html templates `nidmviewer.utils.`**`download_file`**(*src*, *dest*)

`nidmviewer.utils.`**`get_extension`**(*path*)

`nidmviewer.utils.`**`get_images`**(*peaks*, *location_key*)
get_images returns unique images for a location key from a peaks table

`nidmviewer.utils.`**`get_name`**(*path*)

`nidmviewer.utils.`**`get_package_dir`**()

`nidmviewer.utils.`**`get_random_name`**(*length=6*, *chars='ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'*)

`nidmviewer.utils.`**`get_standard_brain`**(*load=True*)

`nidmviewer.utils.`**`is_empty`**(*nii_file*)

`nidmviewer.utils.`**`make_dir`**(*directory*)

`nidmviewer.utils.`**`make_png_paths`**(*nifti_files*)

`nidmviewer.utils.`**`make_tmp_folder`**(**args*, ***kwds*)

`nidmviewer.utils.`**`read_file_lines`**(*file_name*)

`nidmviewer.utils.`**`strip_url`**(*url*, *encode=True*)

`nidmviewer.utils.`**`unwrap_list_unique`**(*list_of_lists*)

`nidmviewer.utils.`**`unzip`**(*source*, *dest_dir*)

2.1.9 nidmviewer.viewer module

viewer.py: part of the nidmviewer package `nidmviewer.viewer.`**`check_inputs`**(*ttl_files*)

check_input. will return a list of ttl files, more advanced functionality can be added if needed (validation, etc) Parameters ========== ttl_files: string or list

> full paths to ttl files
>
> list of ttl files

`nidmviewer.viewer.`**`generate`**(*ttl_files*, *base_image=None*, *retrieve=False*, *view_in_browser=False*, *columns_to_remove=None*, *template_choice='index'*, *port=None*, *remove_scripts=None*, *button_text='BRAIN'*, *check_empty=False*)

will generate a nidmviewer to run locally or to embed into webserver Parameters ========== ttl_files: str or list

> one or more turtle files to add to the viewer. Images in the files should be available at the specified URL.

retrieve: boolean If set to False, the images are assumed to be on the same server, and will be served with the given URL. If retrieve is set to True, the images will be retrieved first and stored in a temporary directory.

base_image: str The base image to use for the viewer. Not specifying a base_image will yield a black background.

view_in_browser: boolean open a temporary web browser (to run locally). If True, images will be copied to a temp folder. If False, image_paths must be relative to web server. File names should be unique.

columns_to_remove: additional columns to remove. If none, default columns of "coordinate_id" "statmap_type" and "exc_set" will be removed.

port: int port to serve nidmviewer, if view_in_browser==True

remove_scripts: list one or more script or button tags to remove from the template. Options include JQUERY BOOTSTRAPJS BOOTSTRAPCSS PAPAYACSS PAPAYAJS NIDMSELECTBUTTON

button_text: str Text string for the button to select a brain image. Default is "BRAIN"

check_empty: boolean - check for empty images or not. Will result in error if nidm paths are URLS.

`nidmviewer.viewer.`**`generate_temp`**(*peaks*, *location_key*)

generate a lookup of temporary files Parameters ========== peaks: dict

> data structure from get_coordinates_and_peaks

location_key: str key in peaks data structure for file paths

peaks: dict (key is ttl file, equivalent to peaks, but old location_key path is replaced with path to temporary directory

copy_list: dict keys are current paths, values are temporary file names corresponding to fullpath in new_nifti_files[ttl_file] dictionary. This is used to copy images into the temporary directory with the correct names.

`nidmviewer.viewer.`**`get_column_names`**(*peaks*)

`nidmviewer.viewer.`**`parse_nidm`**(*ttl_files*)

Extract brainmaps and coordinates from ttl files Parameters: ========== ttl_files: list

list of full paths to ttl files

peaks: dict dict of pandas data frames, one for each ttl_file, with columns coordinate, z, peak_name, pvalue_uncorrected

maps: dict dict of pandas data frames, one for each ttl_file, with columns filename and location for all brain maps specified in ttl.

`nidmviewer.viewer.`**`remove_columns`**(*columns*, *columns_to_remove*)

`nidmviewer.viewer.`**`retrieve_nifti`**(*peaks*, *retrieve*, *location_key*)

Download the image to a temporary folder if the user needs to retrieve it. Otherwise, return file Parameters ========== peaks: dict

dictionary (key, is ttl_file, and value, is dictionary of {filename:fullpath} for all brainmaps extracted from the ttl files

retrieve: boolean if True, will download brainmaps to temporary folder first. If false, encodes path in utf-8 for rendering in javascript

location_key: str key to look up file name in peaks dictionary

`nidmviewer.viewer.`**`to_dictionary`**(*df*, *orient='records'*, *strip_columns=False*, *strings=False*)

to_dictionary: Convert a pandas dataframe into the string of a json/dict to embed into page Parameters ========== df: pandas data frame

data frame to convert

orient: str orientation to convert with (default is “records”)

strip_columns: boolean if true, will return df.to_dict(orient=orient).values() default is False

strings: boolean True will convert all columns to strings.

2.1.10 Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

nidmviewer, 11
nidmviewer.browser, 5
nidmviewer.convert, 6
nidmviewer.scripts, 6
nidmviewer.sparql, 7
nidmviewer.templates, 8
nidmviewer.utils, 8
nidmviewer.viewer, 9

Index

A

add_javascript_function() (in module nidmviewer.templates), 8
add_string() (in module nidmviewer.templates), 8

C

check_inputs() (in module nidmviewer.viewer), 10

D

do_query() (in module nidmviewer.sparql), 7
download_file() (in module nidmviewer.utils), 9

G

generate() (in module nidmviewer.viewer), 10
generate_temp() (in module nidmviewer.viewer), 10
get_column_names() (in module nidmviewer.viewer), 10
get_coordinates() (in module nidmviewer.sparql), 7
get_coordinates_and_maps() (in module nidmviewer.sparql), 7
get_extension() (in module nidmviewer.utils), 9
get_image() (in module nidmviewer.templates), 8
get_images() (in module nidmviewer.utils), 9
get_name() (in module nidmviewer.utils), 9
get_package_dir() (in module nidmviewer.utils), 9
get_random_name() (in module nidmviewer.utils), 9
get_standard_brain() (in module nidmviewer.utils), 9
get_svg_html() (in module nidmviewer.browser), 6
get_template() (in module nidmviewer.templates), 8
getjson() (in module nidmviewer.convert), 6

I

internal_view() (in module nidmviewer.browser), 6
is_empty() (in module nidmviewer.utils), 9

M

main() (in module nidmviewer.scripts), 7
make_dir() (in module nidmviewer.utils), 9
make_png_paths() (in module nidmviewer.utils), 9
make_tmp_folder() (in module nidmviewer.utils), 9

N

nidmviewer (module), 11
nidmviewer.browser (module), 5
nidmviewer.convert (module), 6
nidmviewer.scripts (module), 6
nidmviewer.sparql (module), 7
nidmviewer.templates (module), 8
nidmviewer.utils (module), 8
nidmviewer.viewer (module), 9

P

parse_coordinates() (in module nidmviewer.convert), 6
parse_nidm() (in module nidmviewer.viewer), 11
prettyjson() (in module nidmviewer.convert), 6

R

read_file_lines() (in module nidmviewer.utils), 9
read_template() (in module nidmviewer.templates), 8
remove_columns() (in module nidmviewer.viewer), 11
remove_resources() (in module nidmviewer.templates), 8
retrieve_nifti() (in module nidmviewer.viewer), 11
run_webserver() (in module nidmviewer.browser), 6

S

save_template() (in module nidmviewer.templates), 8
strip_url() (in module nidmviewer.utils), 9

T

to_dictionary() (in module nidmviewer.viewer), 11

U

unwrap_list_unique() (in module nidmviewer.utils), 9
unzip() (in module nidmviewer.utils), 9

V

view() (in module nidmviewer.browser), 6

W

write_file() (in module nidmviewer.browser), 6